1 Dataset

For this proyect, the dataset chosen was *Combined Cycle Power Plant Data Set*. According to the datasets information:

> A combined cycle power plant is composed of gas turbines, steam turbines and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is colected from and has effect on the Steam Turbine, he other three of the ambient variables effect the GT performance.

My main motivation to use this dataset was that the last month there was a kaggle challenge (ASHRAE - Great Energy Predictor III) that tried to answer the question: 'How much energy will a building consume?'. I think this kind of problems dealing with the estimation of energy production and consumption will start to appear more and more and I personally find them to be both really interesting and challenging.

Furthermore, I think that the data was complicated enough to challenge common regression techniques yet simple enough to be solved without advanced domain knowledge or the requirement of more advanced modelling habilities.

2 Features

The dataset contains 9568 observations collected from a CCPP over the period 2006-2011 and the features consist of:

Feature name	Min	Max	Data type
Temperature (AT)	1.81 °C	37.11°C	Continous
Ambient Pressure (AP)	992.89 mb	1033.30 mb	Continous
Relative Humidity (RH)	25.56%	100.16%	Continous
Exhaust Vacuum (V)	25.36 cm Hg	81.56 cm Hg	Continous
Electrical energy output (PE)	420.26 MW	495.76 MW	Continous

The individual distribution of each feature is shown in the following figure:

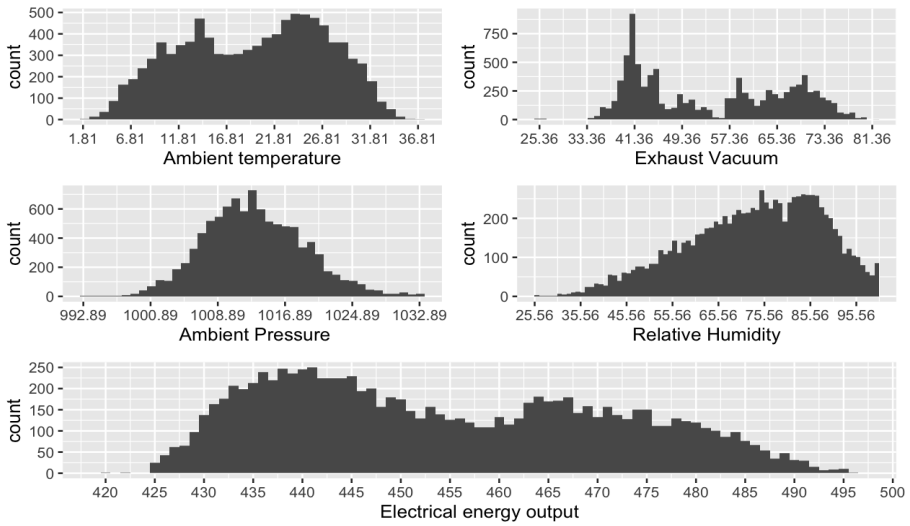


Figure 1: Histogram of the variables

Both the ambient temperature and the electrical energy output seem to have a bimodal distribution, the relative humidity is unimodal with a negative skew, the ambient pressure appears to be normally distributed and the exhaust vacuum shows the most complicated distribution with several distinct modes.

3 Relations of the predictors with the response variable

First, let's take a look to the correlation matrix:

	PE	AP	RH	V	AT
PE	1				
AP	.52	1			
RH	.39	.1	1		
V	-.87	-.41	-.31	1	
AT	-.95	-.51	-.54	.84	1

the response variable has a high negative correlation with both the ambient temperature and the exhaust vacuum, also those features are highly correlated between them.

Now, let's focus on the individual relations of the predictors with the response variable:

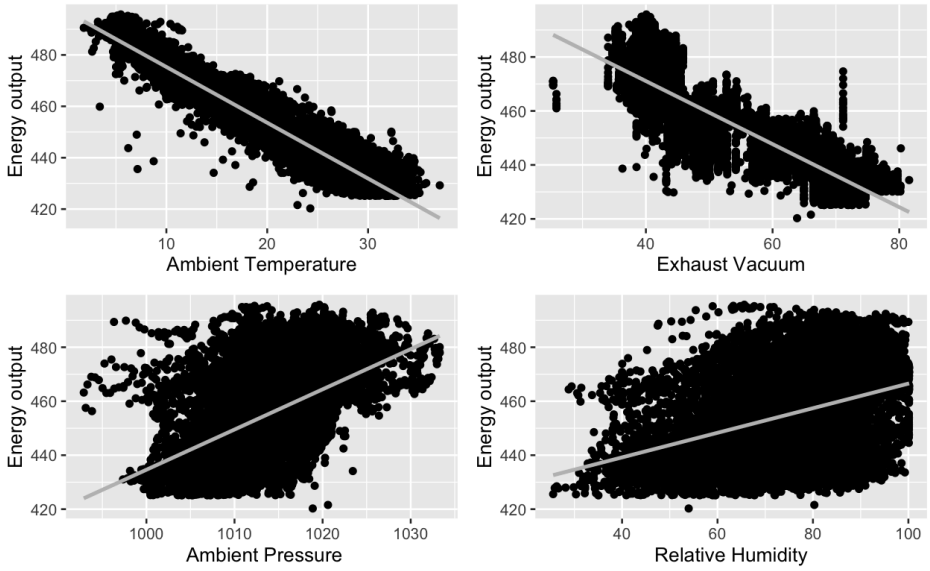


Figure 2: Individual relations of predictors with response

There's a clear relationship between the temperature and the energy output: for a higher temperature, a lower energy output corresponds. Also the exhaust vacuum appear to show a relationship of that kind. We can't extract any conclusions from the ambient pressure and the relative humidity from this figure.

To finish the analysis of our features, it's important to mention that the data have no misssing values and it looks like we don't have to deal with outliers or further cleaning so we'll just standardize each variable and proceed with the modelling.

4 Modelling

All the predictors and the response are continuous variables so the task for this dataset is a regression.

A gradient boosted decision tree was used to solve the regression task since I think it's a good fit for the problem in hand and personally wanted to explore and learn more about this particular method.

A more simple and interpretable model was used as a baseline to compare the performance of the gradient boosting model. A linear regresion was fitted, lasso and ridge methods were tried but multicollinearity weren't a problem and the improvement in terms of test error was minimal so I decided in favor of the more simple linear regresion.

5 Software

Tha programming language used for the analysis of the CCPP dataset was R and the following table describes the main packages and libraries utilized:

Library	Description
Tidyverse	For data manipulation and visualization.
Caret (Classification And Regression Training)	The package goal is to provide a uniform interface to standardize common modelling tasks such parameter tuning and variable importance. In particular, *caret* was used for the cross validation procedure.
XGBoost(Extreme Gradient Boosting)	An optimized distributed library which provides a gradient boosting framework for several programming languages including R.
xgboostExplainer	An R package which allows the predictions from an XGBoost model to be split into the impact of each feature, making the model more interpretable.

6 Linear model

The linear regresion model was straightforward, all the predictors were significant to the response and a significant regression equation was found ($F(4, 7651) = 2.517 \times 10^{-4}, p < 2.2 \times 10^{-16}$) with an $R^2 = 0.9294$.

7 Gradient boosted decision tree

Using the library XGBoost a gradient boosted decision tree model was fitted to predict the electrical energy output. The first issue to solve was how to pick the optimal hyperparameters of the model.

7.1 Parameter tuning

According to XGBoost there are many hyperparameters to be tuned. The following table resumes the most important, it's meaning, and the value used.

Parameter	Description	Value
eta	After each boosting step, eta shrinks the feature weights to make the boosting process more conservative and less likely to overfit	.1
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree. A larger gamma implies a more conservative algorithm.	0
max_depth	Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.	6
min_child _weight	If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.	2

To select, eta, max_depth and min_child_weight a grid search was performed and the optimal values were selected with 5 fold cross-validation using caret *train* function. After selecting the optimal hyperparameters, the optimal number of rounds for boosting was found to be 600.

7.2 Model training and evaluation

With those parameters, the *xgb.train* function was used to train the optimal model. The test mean squared error was 0.032 and, as seen in the predicted vs actual energy output plot the predictions are really good.

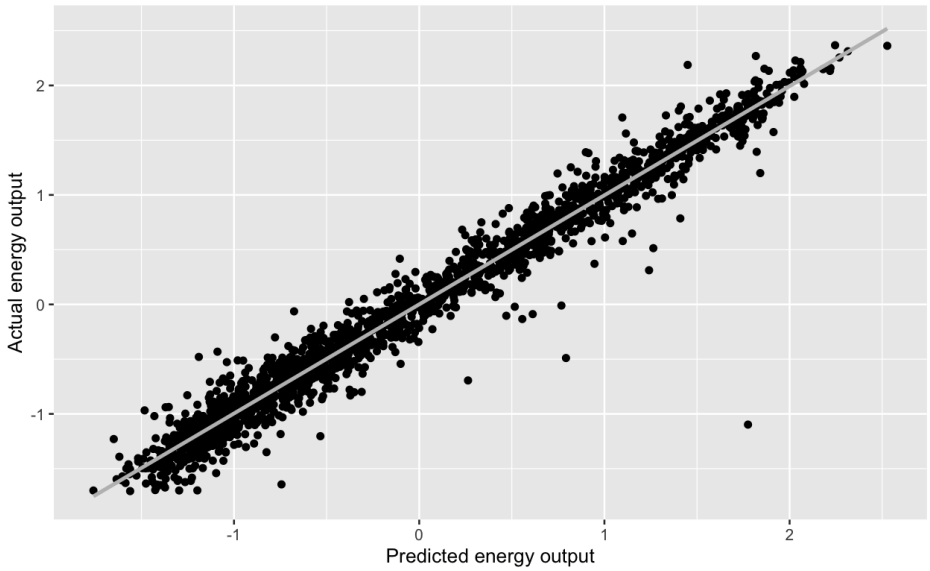


Figure 3: Actual vs Predicted energy output

7.3 Feature importance

After the boosted trees are constructed, we can build an importance score for each attribute. That score indicates how valuable each feature was in the fitting process. With the function *xgb.importance*, the following importance values were calculated.

Library	Description
Ambient temperature	.90
Exhaust Vacuum	.06
Ambient Preassure	.015
Relative Humidity	.013

The ambient temperature is by far the most important feature, followed by the exhaust vacuum. The ambient preassure and the relative humidity weren't very relevant.

7.4 Interpretation

Using the The *xgboostExplainer* R package, we can extract the log-odds contribution of each feature in a prediction by adding up the contributions of each one of them for every tree in the ensemble. With that information, we can measure the impact that each feature had in a particular prediction.

Each plot in the following figure shows the values of a predictor plotted against the impact associated with that value.

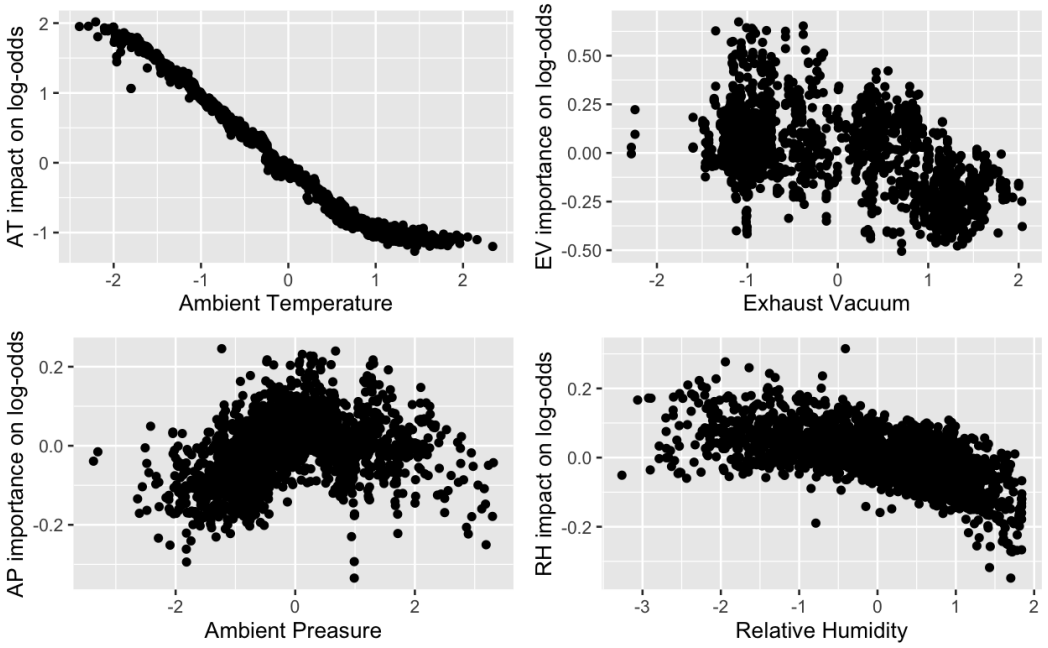


Figure 4: Value vs impact

In the first plot we can see that for lower temperatures the impact is positive (lower temperature contributes to a greater energy output) and as the temperature increases, the impact decreases (a greater temperature contributes to a smaller energy output)

The plot of ambient preassure suggests that, for smaller and bigger values of the preassure the impact to the response is negative (lower and higher preasures lead to an smaller energy output) whilst a mean preassure has, in general, a positive impact (a mean preassure leads to a greater energy output).

The plots of exhaust vacuum and relative humidity will be analized with the following graph where each dot is colored with red if the temperature was higher than the mean and blue if it was lower than the mean.

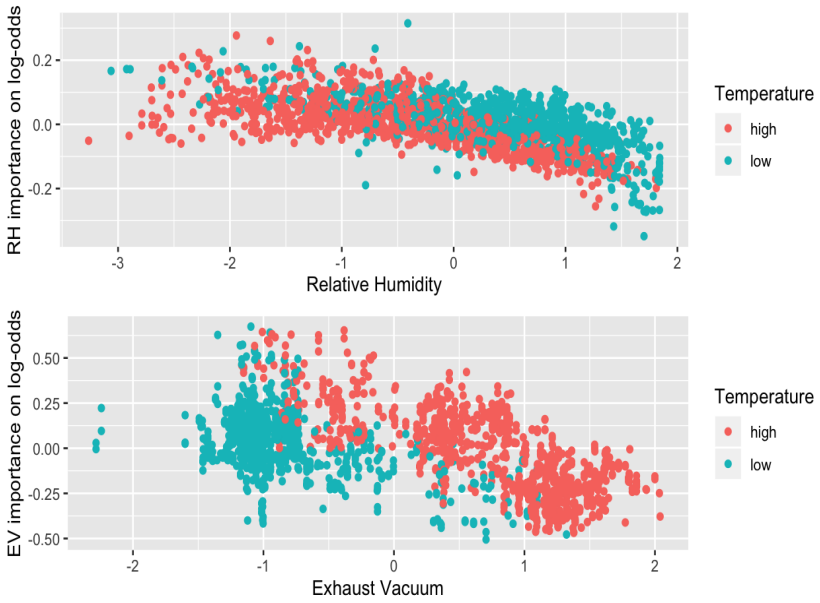


Figure 5: Value vs impact

From the first plot we can see that, for bigger values of humidity we may observe greater energy outputs if the temperature is above the mean; and smaller energy outputs if the temperature is below the mean. A similar analysis can be made for the second plot.

8 Conclusions

The linear model achieved good results and is definitely a good alternative to analyze the problem since it's the most interpretable model we could use.

The gradient boosted decision tree model was able to capture some nonlinear relationships between the predictors achieving a mean squared error 76% smaller than the one from the linear model.

Since the gradient boosting is a complicated model, usually the results lack interpretation. Nevertheless, there exist tools to helps us extract important information about the way in which the model is working and how the predictions are made.

I'm really satisfied with the fact that the gradient boosting model achieves good prediction power without renouncing to interpretation. The gradient boosting framework is, indeed, a very powerfull one.

Posible improvements might involve be in several areas:

- The dataset only uses the 4 more common variables in a CCPP, more features could be added for a broader analysis.
- The results of both the linear and the gradient boosting models could be further interpreted, maybe with more work and domain knowledge.

- Only the main parameters of the gradient boosting model were tuned, we could try doing more exhaustive hyperparameter optimization
- Other regression techniques could be tried in order to achieve better predictions or further insights.

Appendix (Code)

Sergio Arnaud

10 january, 2019

```
library(tidyverse)

library(corrplot)
require(gridExtra)

library(caret)
library(xgboost)

library(xgboostExplainer) #install_github("AppliedDataSciencePartners/xgboostExplainer")
```

Reading and exploring the data

Reading data

```
data <- read_csv('CCPP/data.csv')
```

```
## Parsed with column specification:
## cols(
##   AT = col_double(),
##   V = col_double(),
##   AP = col_double(),
##   RH = col_double(),
##   PE = col_double()
## )
```

```
summary(data)
```

```
##        AT              V               AP              RH
##  Min.   : 1.81   Min.   :25.36   Min.   : 992.9   Min.   : 25.56
##  1st Qu.:13.51   1st Qu.:41.74   1st Qu.:1009.1   1st Qu.: 63.33
##  Median :20.34   Median :52.08   Median :1012.9   Median : 74.97
##  Mean   :19.65   Mean   :54.31   Mean   :1013.3   Mean   : 73.31
##  3rd Qu.:25.72   3rd Qu.:66.54   3rd Qu.:1017.3   3rd Qu.: 84.83
##  Max.   :37.11   Max.   :81.56   Max.   :1033.3   Max.   :100.16
##        PE
##  Min.   :420.3
##  1st Qu.:439.8
##  Median :451.6
##  Mean   :454.4
##  3rd Qu.:468.4
##  Max.   :495.8
```

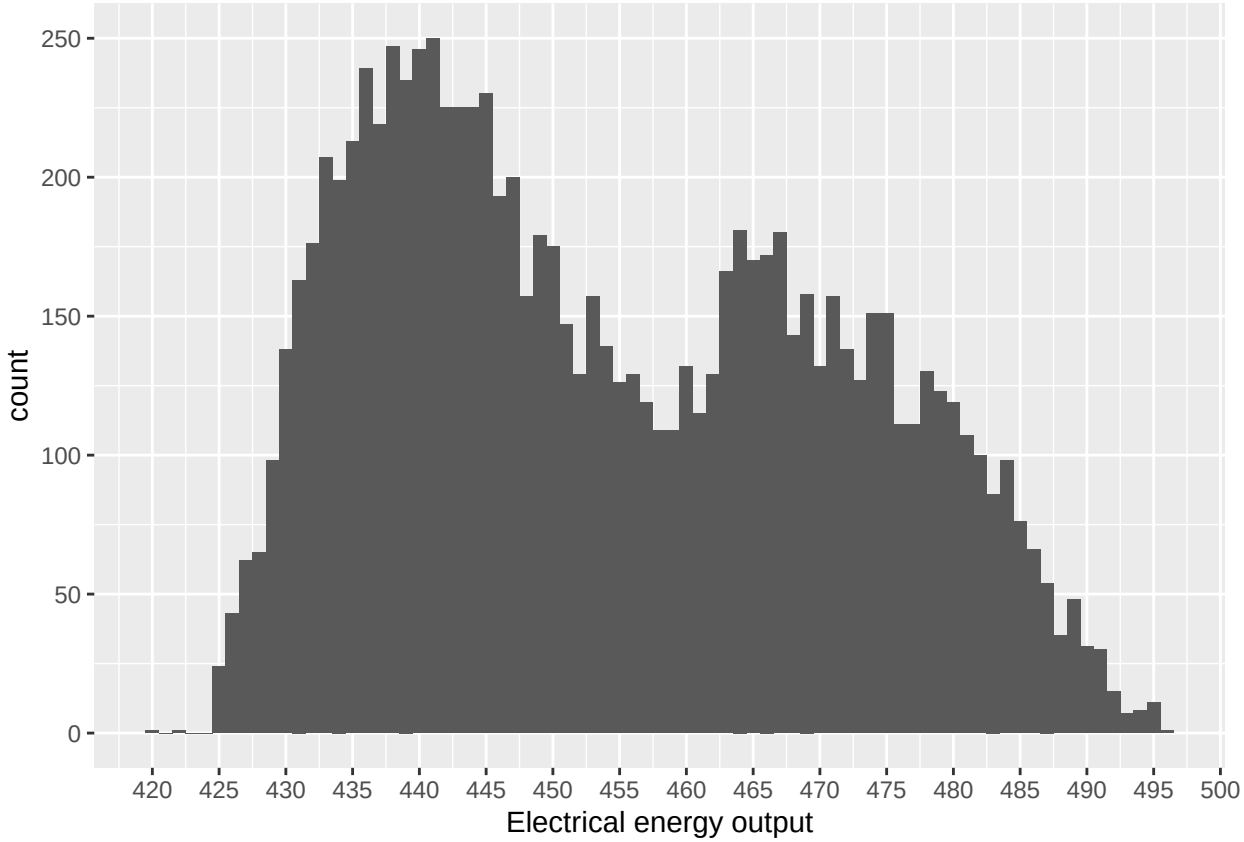
The data dimensions are 9568,5. We have 1231 observations of 5 variables which are: - Temperature (T) - Ambient Pressure (AP) - Relative Humidity (RH) - Exhaust Vacuum (V) - Electrical energy output (EP)

Each variable is continous and the Electrical energy output is the variable to predict.

Variables

The response variable PE, the net hourly electrical energy output of the plant has the following distribution.

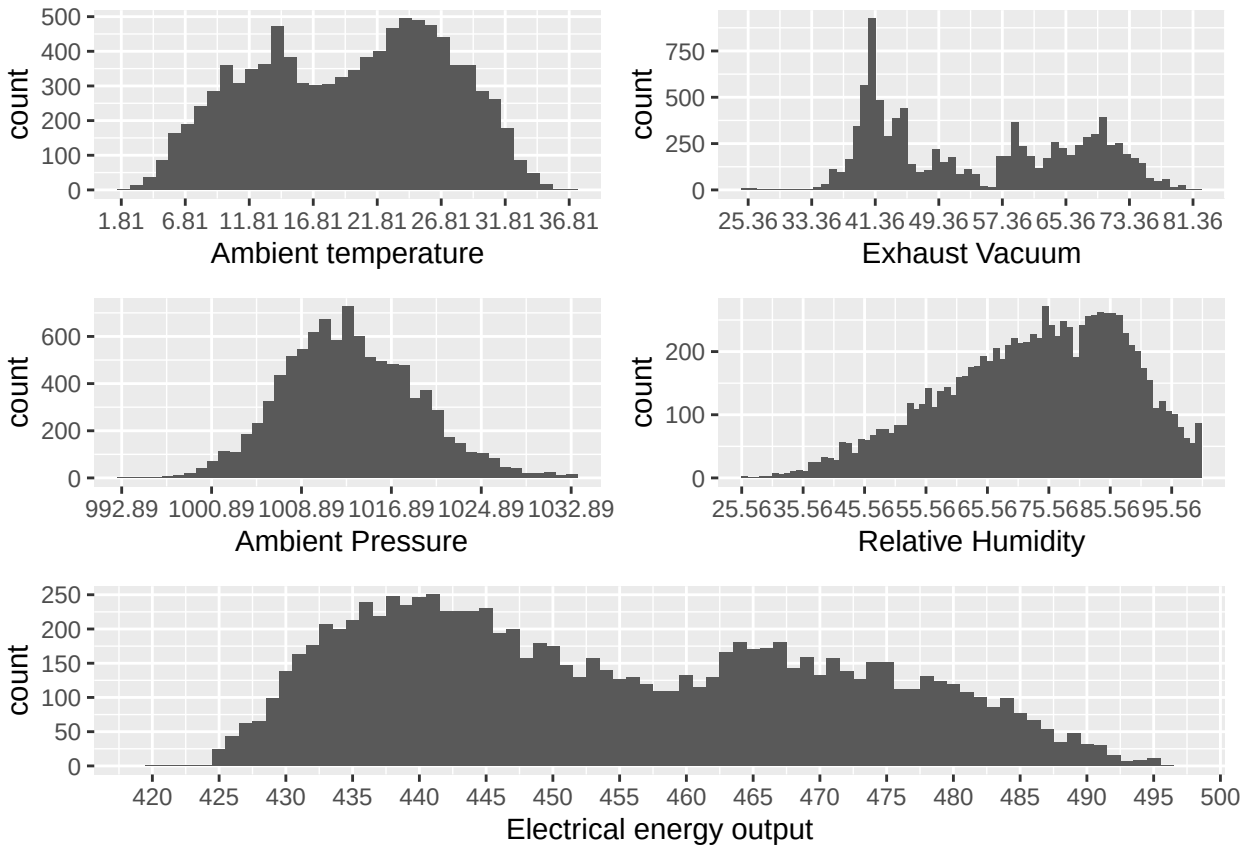
```
ggplot(data=data, aes(x=PE)) +
        geom_histogram(binwidth = 1) +
        scale_x_continuous(breaks= seq(410, 500, by=5)) +
        xlab('Electrical energy output')
```



And the distributions of the 5 variables are

```
p1 <- ggplot(data=data, aes(x=AT)) +
      geom_histogram(binwidth = 1) +
      scale_x_continuous(breaks= seq(min(data$AT), max(data$AT), by=5))+
      xlab('Ambient temperature')

p2 <- ggplot(data=data, aes(x=V)) +
      geom_histogram(binwidth = 1) +
      scale_x_continuous(breaks= seq(min(data$V), max(data$V), by=8))+
      xlab('Exhaust Vacuum')

p3 <- ggplot(data=data, aes(x=AP)) +
      geom_histogram(binwidth = 1) +
      scale_x_continuous(breaks= seq(min(data$AP), max(data$AP), by=8))+
      xlab(' Ambient Pressure')

p4 <- ggplot(data=data, aes(x=RH)) +
      geom_histogram(binwidth = 1) +
      scale_x_continuous(breaks= seq(min(data$RH), max(data$RH), by=10))+
```

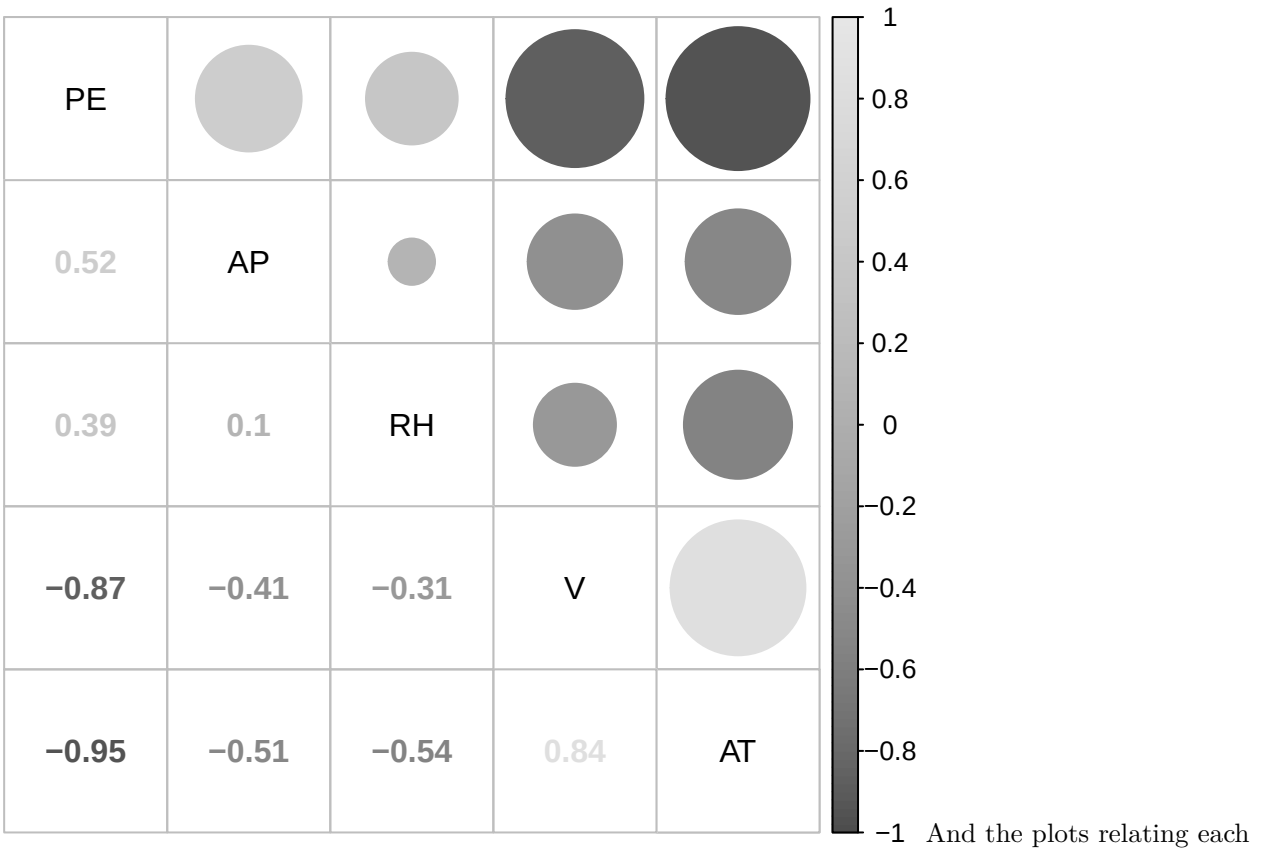
```
      xlab('Relative Humidity')

p5 <- ggplot(data=data, aes(x=PE)) +
        geom_histogram(binwidth = 1) +
        scale_x_continuous(breaks= seq(410, 500, by=5)) +
        xlab('Electrical energy output')

grid.arrange(grobs = list(p1,p2,p3,p4,p5), layout_matrix = rbind(c(1,2),
                                                                 c(3,4),
                                                                 c(5,5)))
```



Relations of the predictors with the response variable

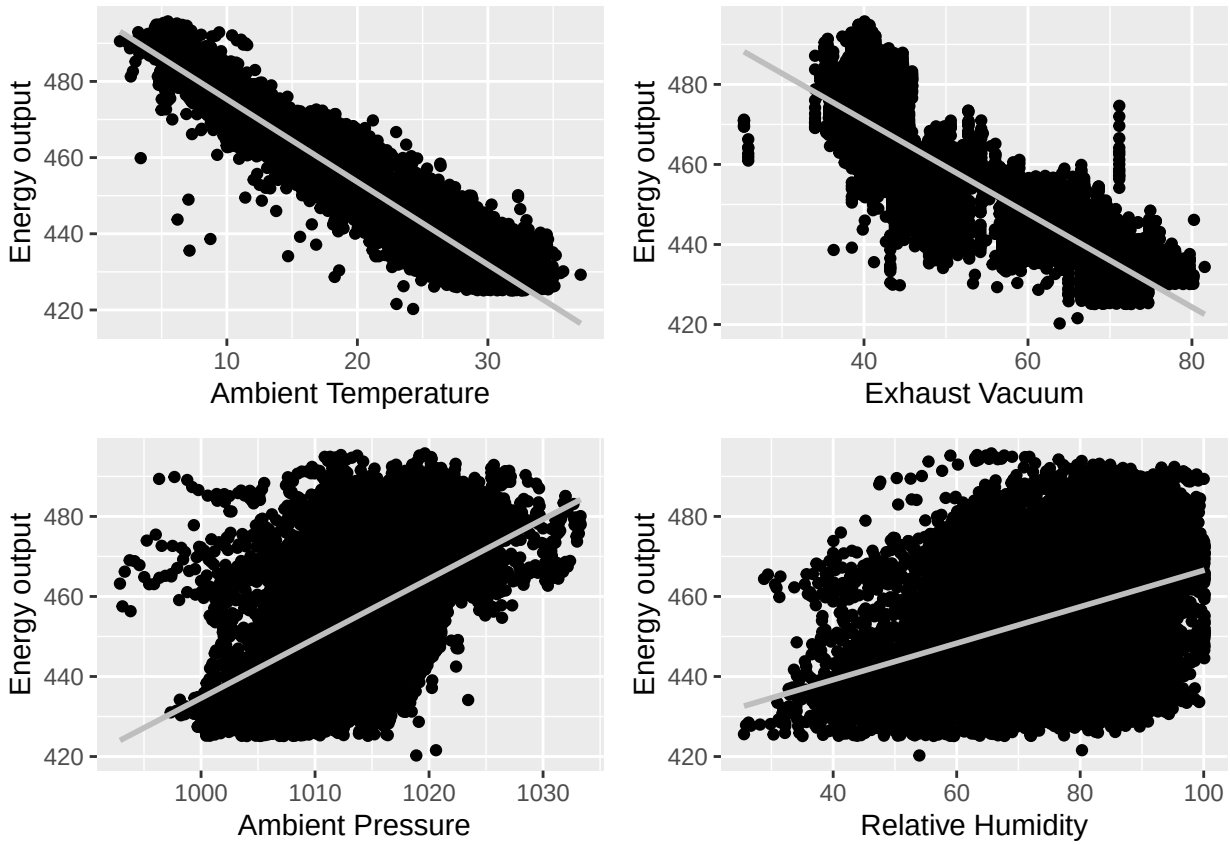
First, let's take a look to the correlation matrix and, in particular, focus in the correlations of each variable with the response variable.

```
correlations <- cor(data, use="pairwise.complete.obs")

cor_sorted <- sort(correlations[,'PE'], decreasing = TRUE)
correlations <- correlations[names(cor_sorted), names(cor_sorted)]

corrplot.mixed(correlations, tl.col="black",
               lower.col = gray.colors(100),
               upper.col = gray.colors(100))
```



And the plots relating each predictor with the response variable

```
p1 <- ggplot(data=data, aes(x=AT, y=PE))+
        geom_point() +
        geom_smooth(method = "lm", se=FALSE, color="grey", aes(group=1)) +
        xlab('Ambient Temperature') +
        ylab('Energy output')

p2 <- ggplot(data=data, aes(x=V, y=PE))+
        geom_point() +
        geom_smooth(method = "lm", se=FALSE, color="grey", aes(group=1)) +
        xlab('Exhaust Vacuum') +
        ylab('Energy output')

p3 <- ggplot(data=data, aes(x=AP, y=PE))+
        geom_point() +
        geom_smooth(method = "lm", se=FALSE, color="grey", aes(group=1)) +
        xlab(' Ambient Pressure') +
        ylab('Energy output')

p4 <- ggplot(data=data, aes(x=RH, y=PE))+
        geom_point() +
        geom_smooth(method = "lm", se=FALSE, color="grey", aes(group=1))+
        xlab('Relative Humidity') +
        ylab('Energy output')

grid.arrange(p1,p2,p3,p4,ncol=2)
```



Preparing data to modelling

Splitting the data in train and test sets

```
set.seed(970628)
trainIndex <- createDataPartition(1:length(data$AT), p=0.8, list=FALSE)

#splitting data into training/testing data using the trainIndex object
train_data <- data[trainIndex,]
test_data <- data[-trainIndex,]

dim(train_data)
```

```
## [1] 7656    5
```

```
dim(test_data)
```

```
## [1] 1912    5
```

```
train_data
```

```
## # A tibble: 7,656 x 5
##       AT     V    AP    RH    PE
##    <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 15.0   41.8 1024.  73.2  463.
##  2 25.2   63.0 1020.  59.1  444.
##  3  5.11  39.4 1012.  92.1  489.
##  4 20.9   57.3 1010.  76.6  446.
##  5 10.8   37.5 1009.  96.6  474.
##  6 26.3   59.4 1012.  58.8  444.
```

```
##  7 15.9   44.0 1014.  75.2  467.
##  8 14.6   45   1022.  41.2  476.
##  9 11.7   43.6 1015.  70.7  478.
## 10 20.1   46.9 1015.  64.2  454.
## # ... with 7,646 more rows
```

It's important to underline that the data have no misssing values and it looks like we don't have to deal with outliers or further cleaning the data, well just standardise each variable.

```
col_mean <- map(train_data, mean)
col_sd <- map(train_data, sd)

train_data <- train_data %>%
    map2_df(col_mean, ~.x - .y) %>% # Remove mean
    map2_df(col_sd, ~.x / .y) # Divide by sd

test_data <- test_data %>%
    map2_df(col_mean, ~.x - .y) %>%
    map2_df(col_sd, ~.x / .y)
```

Linear model

```
lin_model <- lm(data = train_data, PE ~ .)
summary(lin_model)
```

```
##
## Call:
## lm(formula = PE ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.54162 -0.18501 -0.00749  0.18557  1.03778
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -3.644e-16  3.038e-03    0.000        1
## AT          -8.596e-01  7.471e-03 -115.053  < 2e-16 ***
## V           -1.766e-01  6.079e-03  -29.057  < 2e-16 ***
## AP           2.459e-02  3.654e-03    6.729 1.83e-11 ***
## RH          -1.347e-01  3.959e-03  -34.020  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2658 on 7651 degrees of freedom
## Multiple R-squared:  0.9294, Adjusted R-squared:  0.9293
## F-statistic: 2.517e+04 on 4 and 7651 DF,  p-value: < 2.2e-16
```

And the mean square error is

```
ans_linear <- predict(lin_model, test_data %>% select(-PE))
ans_linear <- test_data %>% mutate(real = PE, estimated = ans_linear) %>% select(real, estimated)
ans_linear %>% mutate(mse = (real-estimated)^2) %>% summarise(mean(mse))
```

```
## # A tibble: 1 x 1
##   `mean(mse)`
```

```
##          <dbl>
## 1       0.0726
```

Lasso

Cross validation to fix the lambda

```
my_control <- trainControl(method="cv", number=5)
lasso_grid <- expand.grid(
                          alpha = 1,  # alpha = 1 lasso, # alpha = 0 ridge
                          lambda = seq(0.001,1,by = 0.001)
                          )

lasso_model <- train(x = train_data %>% select(-PE),
                     y = train_data$PE,
                     method='glmnet',
                     trControl = my_control,
                     tuneGrid = lasso_grid)
lasso_model$bestTune
```

```
##   alpha lambda
## 2     1  0.002
```

Mean squared erro

```
predictions_lasso <- predict(lasso_model, test_data %>% select(-PE))
ans_lasso <- test_data %>% mutate(real = PE, estimated = predictions_lasso) %>% select(real, estimated)
ans_lasso %>% mutate(mse = (real-estimated)^2) %>% summarise(mean(mse))
```

```
## # A tibble: 1 x 1
##   `mean(mse)`
##         <dbl>
## 1      0.0725
```

XGBoost

Caret hyper parameter tuning

```
xgb_grid = expand.grid(
    nrounds = 1000,
    eta = c(0.3, 0.2, 0.1),
    max_depth = c(3, 4, 5, 6),
    gamma = 0,
    colsample_bytree=1,
    min_child_weight=c(1, 2, 3, 4),
    subsample=1
)

xgb_caret <- train(x = train_data %>% select(-PE),
                     y = train_data$PE,
                     method='xgbTree',
                     trControl = my_control,
                     tuneGrid = xgb_grid)
```

The results are

```
head(xgb_caret$results)
```

```
##    eta max_depth gamma colsample_bytree min_child_weight subsample nrounds
## 1  0.1         3     0                1                1         1    1000
## 2  0.1         3     0                1                2         1    1000
## 3  0.1         3     0                1                3         1    1000
## 4  0.1         3     0                1                4         1    1000
## 17 0.2         3     0                1                1         1    1000
## 18 0.2         3     0                1                2         1    1000
##         RMSE  Rsquared       MAE      RMSESD  RsquaredSD       MAESD
## 1  0.1902446 0.9638717 0.1374837 0.003953519 0.002244040 0.001313633
## 2  0.1897812 0.9640582 0.1375092 0.003958975 0.002161389 0.001028581
## 3  0.1901313 0.9639191 0.1377980 0.004045142 0.002257899 0.002071299
## 4  0.1909614 0.9636070 0.1387045 0.003813614 0.002164421 0.002170217
## 17 0.1845860 0.9659834 0.1306439 0.003648292 0.001934761 0.001810661
## 18 0.1838510 0.9662304 0.1309183 0.005428448 0.002561988 0.002259691
```

And the best tune for the hyperparameters is

```
xgb_caret$bestTune
```

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 13    1000         6 0.1     0                1                1         1
```

Now let us find the ideal number of rounds

```
label_train <- train_data$PE

# put our testing & training data into two seperates Dmatrixs objects
dtrain <- xgb.DMatrix(data = as.matrix(train_data %>% select(-PE)), label= label_train)
dtest <- xgb.DMatrix(data = as.matrix(test_data %>% select(-PE)))

default_param<-list(
        objective = "reg:squarederror",
        booster = "gbtree",
        eta=0.1,
        gamma=0,
        max_depth=6,
        min_child_weight=2
)

xgbcv <- xgb.cv( params = default_param, data = dtrain, nrounds = 1000, nfold = 5, showsd = T, stratifi
```

```
## [1]  train-rmse:1.011927+0.002215    test-rmse:1.012305+0.009122
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## [41] train-rmse:0.180086+0.002055    test-rmse:0.212726+0.007439
## [81] train-rmse:0.159986+0.001998    test-rmse:0.202507+0.007983
## [121]    train-rmse:0.145730+0.001242    test-rmse:0.195676+0.008700
## [161]    train-rmse:0.133892+0.001974    test-rmse:0.190461+0.008799
## [201]    train-rmse:0.124802+0.001873    test-rmse:0.187532+0.008928
## [241]    train-rmse:0.117295+0.001391    test-rmse:0.185282+0.009086
## [281]    train-rmse:0.110211+0.001545    test-rmse:0.183131+0.009027
## [321]    train-rmse:0.103899+0.001570    test-rmse:0.181547+0.009086
## [361]    train-rmse:0.098591+0.001698    test-rmse:0.180629+0.009304
```

```
## [401]    train-rmse:0.093530+0.001621    test-rmse:0.179902+0.009647
## [441]    train-rmse:0.088497+0.001299    test-rmse:0.178979+0.009841
## [481]    train-rmse:0.084607+0.001161    test-rmse:0.178477+0.009879
## [521]    train-rmse:0.080424+0.000982    test-rmse:0.177907+0.009813
## [561]    train-rmse:0.076793+0.000866    test-rmse:0.177613+0.009912
## [601]    train-rmse:0.073512+0.000792    test-rmse:0.177261+0.010111
## Stopping. Best iteration:
## [628]    train-rmse:0.071292+0.000912    test-rmse:0.177089+0.010107
```

The optimal number of rounds for the choice of hyper parameters is 598. Training the model with the optimal hyperparameters

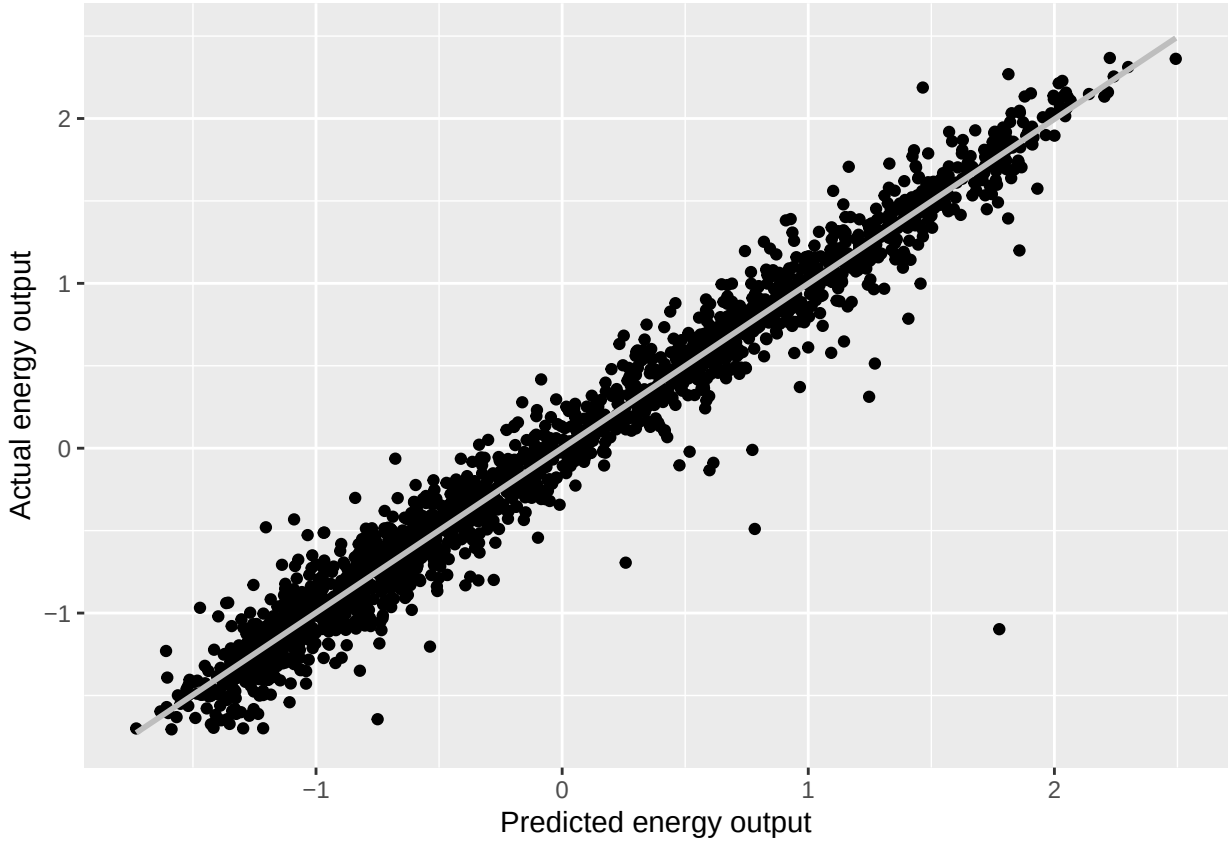
```
xgb_mod <- xgb.train(data = dtrain, params=default_param, nrounds = 598)
```

Predicting and evaluating the prediction results with the test dataset

```
XGBpred <- predict(xgb_mod, dtest)
ans_xgb <- test_data %>% mutate(real = PE, estimated = XGBpred) %>% select(real, estimated)
ans_xgb %>% mutate(mse = (real-estimated)^2) %>% summarise(mean(mse))
```

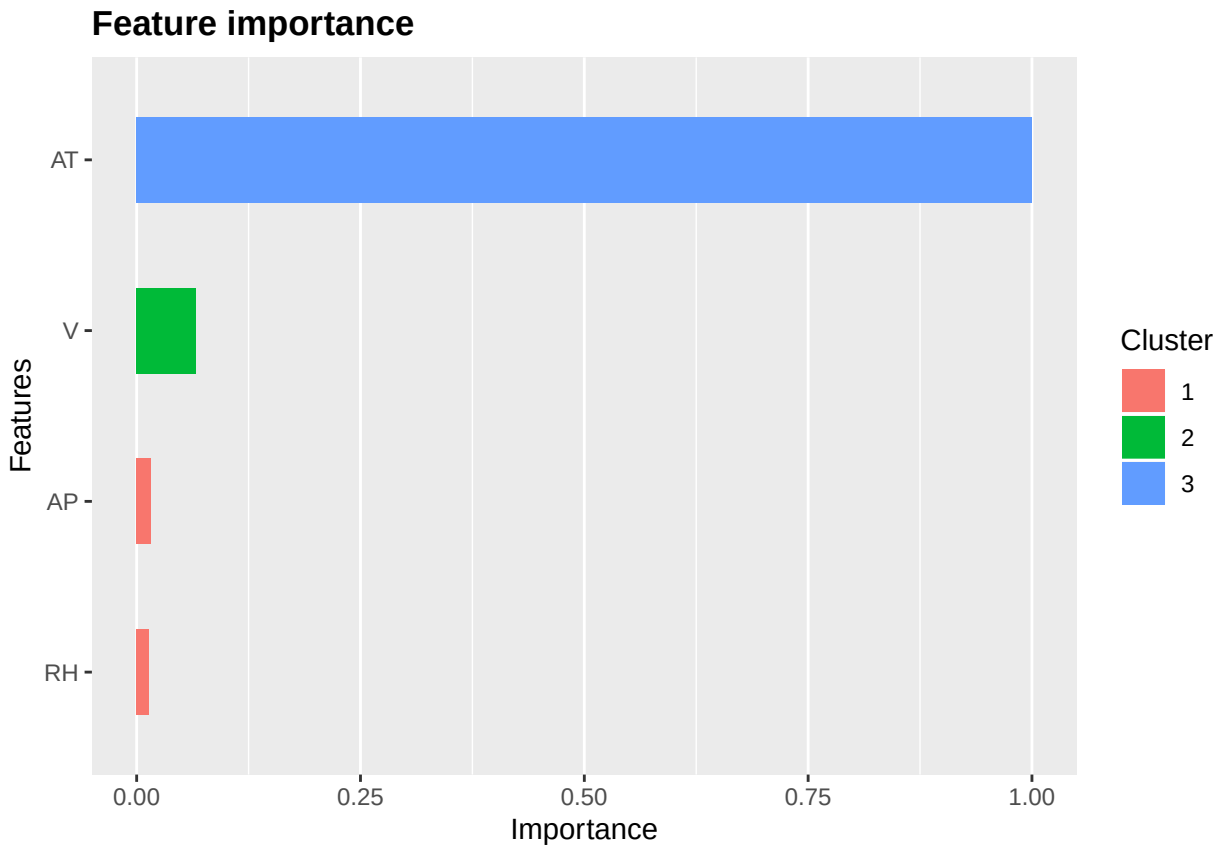
```
## # A tibble: 1 x 1
##   `mean(mse)`
##         <dbl>
## 1      0.0324
```

```
ggplot(data=ans_xgb, aes(x=estimated, y=real))+
        geom_point() +
        geom_smooth(method = "lm", se=FALSE, color="grey", aes(group=1)) +
        ylab('Actual energy output') +
        xlab('Predicted energy output')
```



Plotting the variable importance

```
library(Ckmeans.1d.dp) #required for ggplot clustering
mat <- xgb.importance (feature_names = colnames(train_data),model = xgb_mod)
xgb.ggplot.importance(importance_matrix = mat, rel_to_first = TRUE)
```



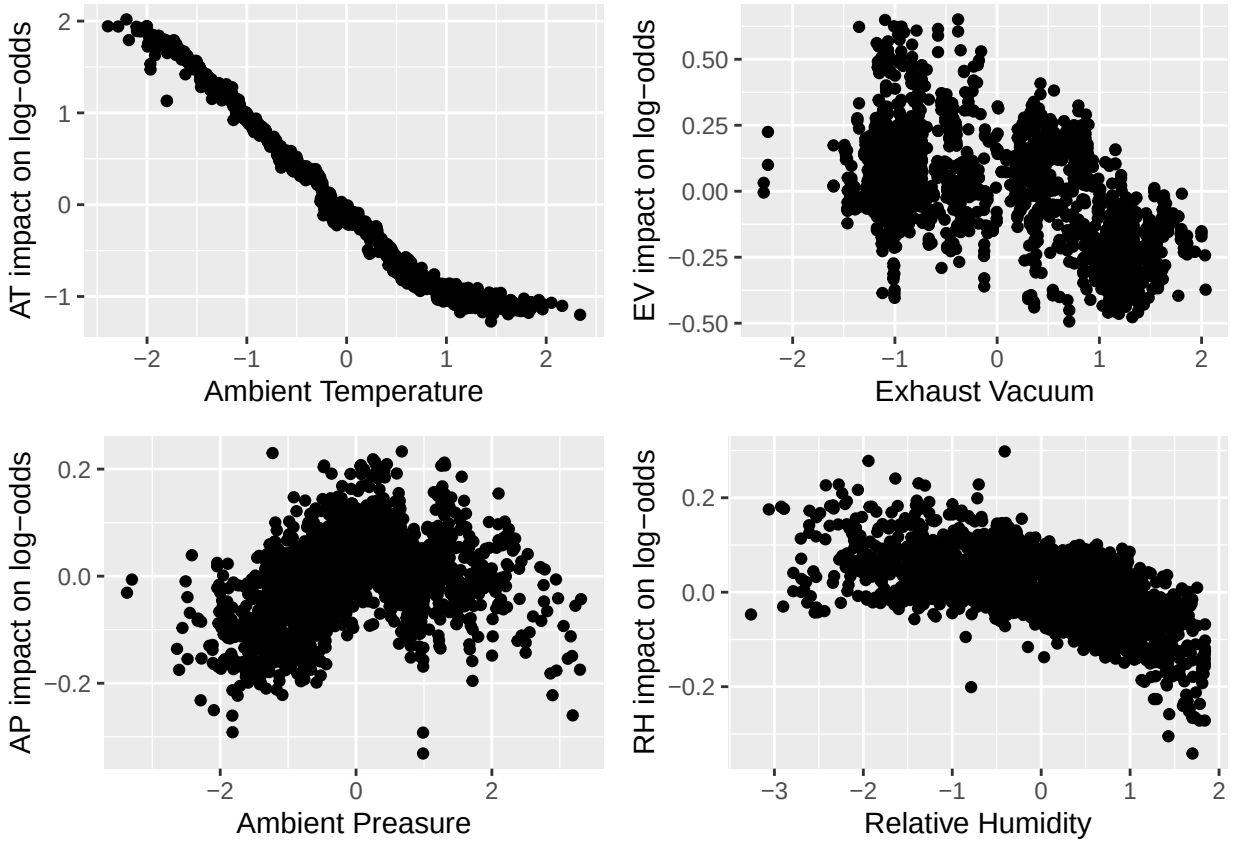
Building explainer of results

```
explainer = buildExplainer(xgb_mod, dtrain, type="binary", base_score = 0.5, trees_idx = NULL)
pred.breakdown = explainPredictions(xgb_mod, explainer, dtest)
colnames(pred.breakdown) <- paste("pred", colnames(pred.breakdown), sep = "_")
```

Each plot in the following figure shows the values of a feature plotted against the impact associated with that value.

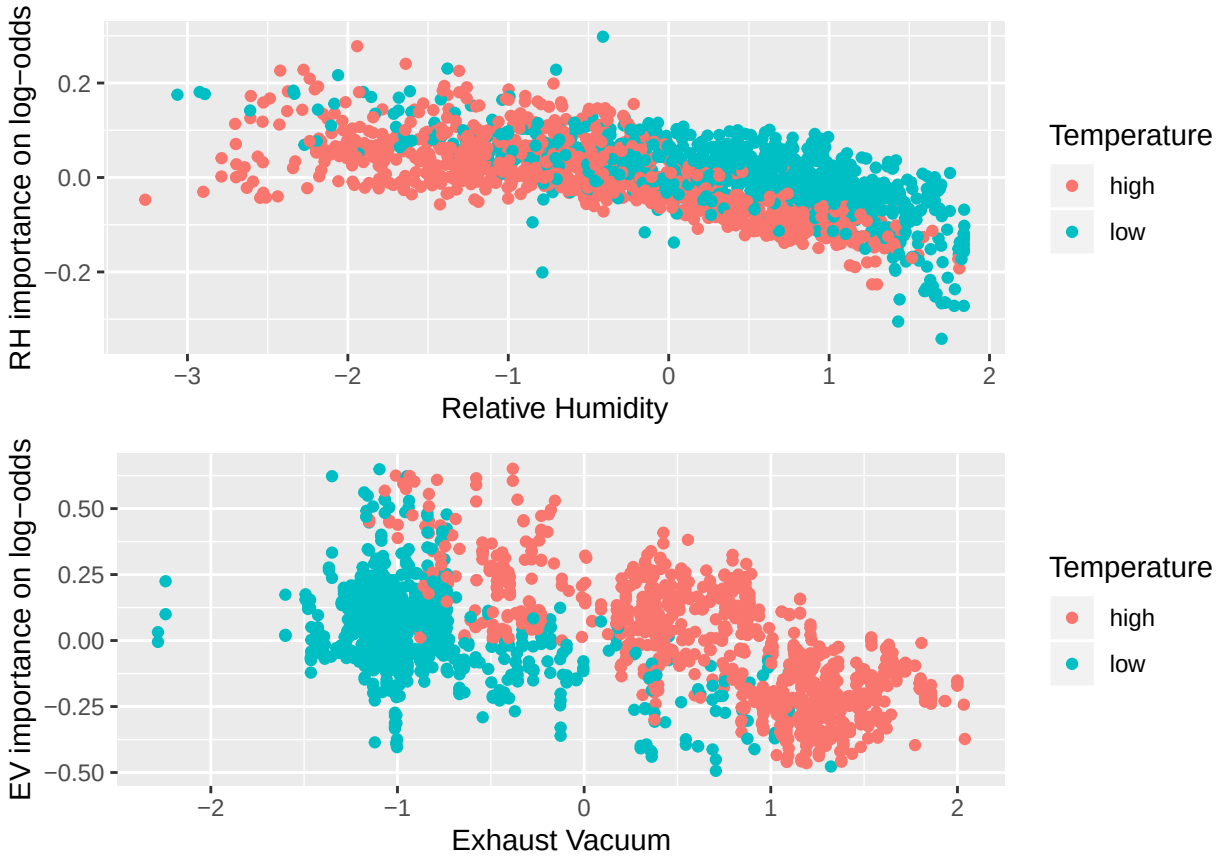
```
expl_data <- cbind(test_data, pred.breakdown)
expl_data <- expl_data %>% mutate(Temperature = ifelse(AT > 0, "high", "low"))

p1 <- ggplot(data=expl_data, aes(x=AT, y=pred_AT))+
        geom_point() +
        xlab('Ambient Temperature') +
        ylab('AT impact on log-odds')

p2 <- ggplot(data=expl_data, aes(x=V, y=pred_V))+
        geom_point()  +
        xlab('Exhaust Vacuum') +
        ylab('EV impact on log-odds')

p3 <- ggplot(data=expl_data, aes(x=AP, y=pred_AP))+
        geom_point() +
        xlab('Ambient Preasure') +
        ylab('AP impact on log-odds')

p4 <- ggplot(data=expl_data, aes(x=RH, y=pred_RH))+
```

```
        geom_point() +
        xlab('Relative Humidity') +
        ylab('RH impact on log-odds')

grid.arrange(p1,p2,p3,p4,ncol=2)
```



Each plot in the following figure shows the values of a feature plotted agains the impact associated with that value. Colored by low or high temperature.

```
p1 <- ggplot(data=expl_data, aes(x=V, y=pred_V, col = Temperature))+
        geom_point() +
        xlab('Exhaust Vacuum') +
        ylab('EV importance on log-odds')

p2 <- ggplot(data=expl_data, aes(x=RH, y=pred_RH, col = Temperature))+
        geom_point() +
        xlab('Relative Humidity') +
        ylab('RH importance on log-odds')

grid.arrange(p2,p1,ncol=1)
```



Ilustrating how a single observation is built

```
showWaterfall(xgb_mod, explainer, dtest, test_data,  8, type = "binary")
```

```
##
##
## Extracting the breakdown of each prediction...
##
  |
  |                                                                      |   0%
  |
  |                                                                      |   1%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   2%
  |
  |==                                                                    |   3%
  |
  |==                                                                    |   4%
  |
  |===                                                                   |   4%
  |
  |===                                                                   |   5%
  |
```

```
|====                                                                   |   5%
|
|====                                                                   |   6%
|
|=====                                                                  |   7%
|
|=====                                                                  |   8%
|
|======                                                                 |   8%
|
|======                                                                 |   9%
|
|=======                                                                |   9%
|
|=======                                                                |  10%
|
|=======                                                                |  11%
|
|========                                                               |  11%
|
|========                                                               |  12%
|
|=========                                                              |  12%
|
|=========                                                              |  13%
|
|=========                                                              |  14%
|
|==========                                                             |  14%
|
|==========                                                             |  15%
|
|===========                                                            |  15%
|
|===========                                                            |  16%
|
|============                                                           |  17%
|
|============                                                           |  18%
|
|=============                                                          |  18%
|
|=============                                                          |  19%
|
|==============                                                         |  19%
|
|==============                                                         |  20%
|
|==============                                                         |  21%
|
|===============                                                        |  21%
|
|===============                                                        |  22%
|
```

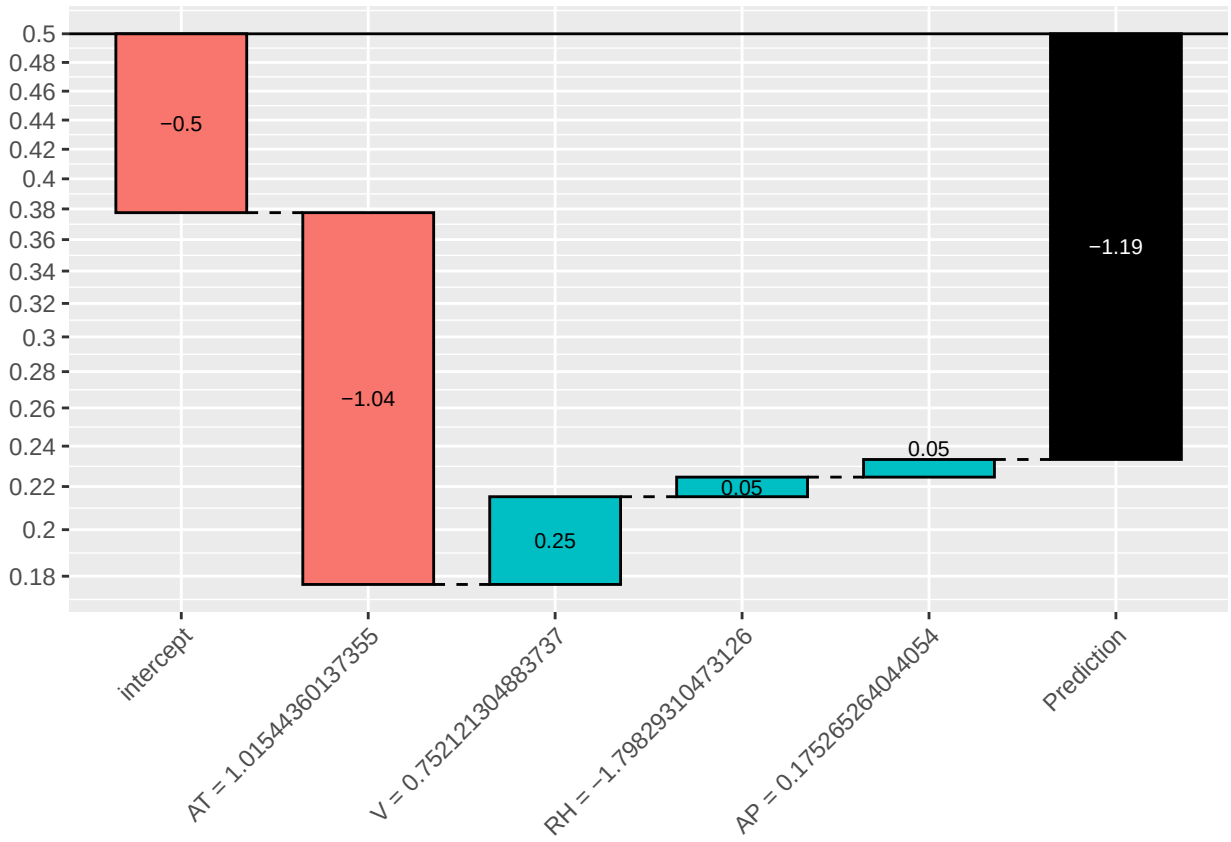
```
|===============                                                  |  22%
|
|================                                                 |  23%
|
|=================                                                |  24%
|
|=================                                                |  25%
|
|==================                                               |  25%
|
|==================                                               |  26%
|
|===================                                              |  27%
|
|===================                                              |  28%
|
|====================                                             |  28%
|
|====================                                             |  29%
|
|=====================                                            |  29%
|
|=====================                                            |  30%
|
|=====================                                            |  31%
|
|======================                                           |  31%
|
|======================                                           |  32%
|
|======================                                           |  32%
|
|======================                                           |  33%
|
|=======================                                          |  34%
|
|=======================                                          |  35%
|
|========================                                         |  35%
|
|========================                                         |  36%
|
|========================                                         |  36%
|
|=========================                                        |  37%
|
|=========================                                        |  38%
|
|==========================                                       |  38%
|
|==========================                                       |  39%
|
|==========================                                       |  39%
|
```

```
|============================                                   |  40%
|
|============================                                   |  41%
|
|============================                                   |  41%
|
|============================                                   |  42%
|
|============================                                   |  42%
|
|============================                                   |  43%
|
|============================                                   |  44%
|
|============================                                   |  45%
|
|============================                                   |  45%
|
|============================                                   |  46%
|
|============================                                   |  46%
|
|============================                                   |  47%
|
|============================                                   |  48%
|
|============================                                   |  48%
|
|============================                                   |  49%
|
|============================                                   |  49%
|
|============================                                   |  50%
|
|============================                                   |  51%
|
|============================                                   |  51%
|
|============================                                   |  52%
|
|============================                                   |  52%
|
|============================                                   |  53%
|
|============================                                   |  54%
|
|============================                                   |  54%
|
|============================                                   |  55%
|
|============================                                   |  55%
|
|============================                                   |  56%
|
```

```
|=====================================                 |  57%
|
|=====================================                 |  58%
|
|======================================                |  58%
|
|======================================                |  59%
|
|=======================================               |  59%
|
|=======================================               |  60%
|
|========================================              |  61%
|
|========================================              |  61%
|
|=========================================             |  62%
|
|=========================================             |  62%
|
|==========================================            |  63%
|
|==========================================            |  64%
|
|===========================================           |  64%
|
|===========================================           |  65%
|
|============================================          |  65%
|
|============================================          |  66%
|
|=============================================         |  67%
|
|=============================================         |  68%
|
|==============================================        |  68%
|
|==============================================        |  69%
|
|===============================================       |  69%
|
|===============================================       |  70%
|
|================================================      |  71%
|
|================================================      |  71%
|
|=================================================     |  72%
|
|=================================================     |  72%
|
|==================================================    |  73%
|
```

```
|================================================                  |  74%
|
|================================================                  |  75%
|
|================================================                  |  75%
|
|================================================                  |  76%
|
|================================================                  |  77%
|
|================================================                  |  78%
|
|================================================                  |  78%
|
|================================================                  |  79%
|
|================================================                  |  79%
|
|================================================                  |  80%
|
|================================================                  |  81%
|
|================================================                  |  81%
|
|================================================                  |  82%
|
|================================================                  |  82%
|
|================================================                  |  83%
|
|================================================                  |  84%
|
|================================================                  |  85%
|
|================================================                  |  85%
|
|================================================                  |  86%
|
|================================================                  |  86%
|
|================================================                  |  87%
|
|================================================                  |  88%
|
|================================================                  |  88%
|
|================================================                  |  89%
|
|================================================                  |  89%
|
|================================================                  |  90%
|
|================================================                  |  91%
|
```

```
  |=================================================================      |  91%
  |
  |=================================================================      |  92%
  |
  |==================================================================     |  92%
  |
  |==================================================================     |  93%
  |
  |===================================================================    |  94%
  |
  |===================================================================    |  95%
  |
  |====================================================================   |  95%
  |
  |====================================================================   |  96%
  |
  |=====================================================================  |  96%
  |
  |=====================================================================  |  97%
  |
  |=====================================================================  |  98%
  |
  |====================================================================== |  98%
  |
  |====================================================================== |  99%
  |
  |=======================================================================|  99%
  |
  |=======================================================================| 100%
##
## DONE!
##
## Prediction:  0.2332554
## Weight:  -1.19002
## Breakdown
##   intercept          AT           V          RH          AP
## -0.49999264 -1.03940021  0.24528357  0.05478898  0.04930068
```


0.5
0.48
0.46
0.44
0.42
0.4
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
0.22
0.2
0.18
−0.5
−1.04
0.25
0.05
0.05
−1.19
intercept
AT = 1.01544360137355
V = 0.752121304883737
RH = −1.79829310473126
AP = 0.175265264044054
Prediction